EZ-ZONE® Integrated Panel Mount Controller EtherNet/IP Configuration & Startup Using an Allen-Bradley CompactLogix PLC

EtherNet/IP Fundamentals

EtherNet/IP is built on the Common Industrial Protocol (CIP) at a foundational level. When communicating using CIP there are two ways to communicate to/from the Master and Slave devices, i.e., Implicitly (real-time I/O messaging) and Explicitly (information/configuration messaging). For your reference, the Watlow device is always the Slave where the PLC is the Master on the network. This document will look closely at both methods of communication.

Explicit Communications - Defined

This type of messaging is executed on demand and can vary in size. Every message must be individually configured to execute a specific Message Type, e.g., CIP Generic and a specific Service Type, e.g., Get Attribute Single. Each device will interpret the message, act upon the task and then generate a response. This message type encapsulates information about the protocol itself as well as the instructions that need to be carried out in a TCP/IP packet. When a message is sent using TCP/IP it requires a response from the device. As stated above, this type of message is generally reserved for diagnostics and configuration.

Implicit Communications - Defined

Because implicit messaging is real-time I/O messaging, it places different demands on the system. Due to the time critical nature of this form of communications the protocol must be able to support multi-casting while also ensuring that the time to execute the task is as fast as possible. To do this effectively, EtherNet/IP incorporates a protocol called User Datagram Protocol/Internet Protocol (UDP). Basically, this protocol contains the data alone without requiring a response from the Slave device. All data that is passed implicitly is defined in the configuration or start up process. Because this method of communications contains the predefined data alone, it is considered to be low overhead and is therefore able to deliver the time-critical requirements for control.

By using both forms of communication EtherNet/IP can prioritize time-critical I/O communications over non-critical messages while allowing for both to occur simultaneously. Watlow EtherNet/IP equipped devices supports both forms (Explicit/Implicit) of communications.

1.0 Getting Started

Prior to configuring the EZ-ZONE PM controller it is important to think through the needs of the application while also understanding some basic facts that pertain to this device.

Note:

This document will not cover basic configuration of the PM controller for this is covered in the PM Integrated (PMI) User's Guide which can be found on the Watlow website; link provided below. http://www.watlow.com/literature/manuals.cfm.

1.1 Noteworthy PM Facts

1.1.1 The PMI controller equipped with the EtherNet/IP communications card is fully tested and compatible with any EtherNet/IP network (no gateways required).

1.1.2 In this documentation, the CIP input assembly is referred to as the Originator to Target (O to T, instance 1) where the output assembly is referred to as the Target to Originator (T to O, instance 2). The Originator is the Master (usually a PLC) and the Target is the Slave (EZ-ZONE PM controllers).

1.1.3 All EZ-ZONE PM assembly members (inputs and outputs) are 32-bits.

1.1.4 The maximum PM implicit assembly size is 20 inputs and 20 outputs.

1.2 Understanding the Application Requirements

1.2.1 Will there be a need to infrequently read or write parameters between the Master and Slave? Explicit communications can be executed with minimal effort to accomplish this task. Setup and configuration can be found below (see: Explicit Communications Configuration Step-by-Step).

1.2.2 If using implicit communications determine what data (EZ-ZONE parameters) will be transferred implicitly (inputs and outputs) between the Master and Slave (20 input and output members maximum). Refer to the PMI User's Guide to find parameters of choice as well as specific data types for each. If not already in hand, click on the link that follows to retrieve this document from the Watlow website:http://www.watlow.com/downloads/en/manuals/pmpmi.pdf

1.2.3 Will the default EZ-ZONE device assemblies meet the application requirements or will the device assembly need to be modified? To answer this question, refer to the User's Guide in the previous step to evaluate the default assemblies for each device.

1.2.4 How fast does the assembly information (I/O) need to be refreshed? When communicating implicitly, the Master (PLC) controls the cyclic timing (I/O updates) via a setting referred to as the RPI.

Note:
Suggested RPI setting should be set between 250 and 500ms.

2.0 Explicit Communications

2.1 Configuration

It should be noted here that if it is determined that the default Implicit Assemblies need to be changed (step 1.2.3 above), this is the communications method to use to accomplish that task. To establish explicit communications between Master and Slave devices configuration steps need to be executed within the PLC as well as the PMI. After the configuration requirements have been met, programming examples will follow.

PM Configuration - Required Steps Using EZ-ZONE Configurator Software

a. Identify the PMI on the Ethernet network via an IP address
b. Enable EtherNet/IP
c. Define the input and output Implicit Assembly sizes (20 inputs and 20 outputs maximum).

PLC Configuration - Required Step Using RSLogix5000 Software

a. Add a Generic Ethernet module to the PLC I/O structure
b. Configure the module properties, e.g., IP address, Assembly size, etc…

2.2 PMI Configuration, Step-by-Step

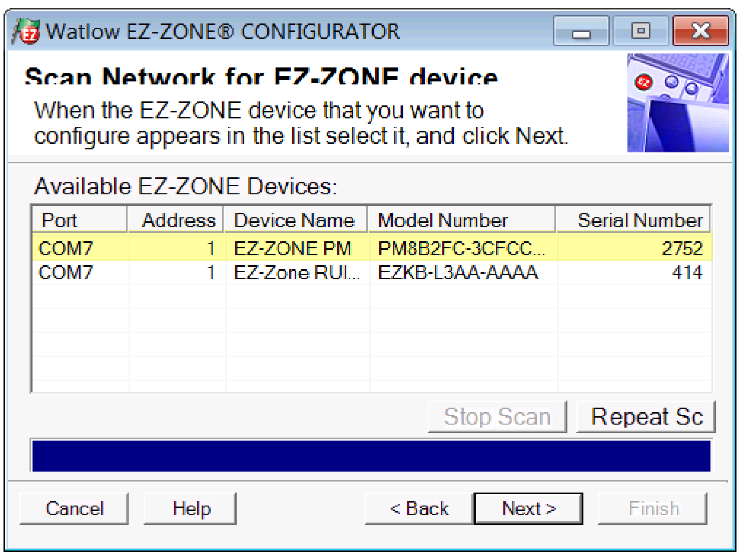
2.2.1 Using an EIA-485 connection using Standard Bus, open up EZ-ZONE Configurator software (EZ-ZONE Configurator does not use Ethernet connection) and configure a device while communicating.

Note:

If more information is needed in connecting the PMI to the PC, review the PMI User's Guide and turn to the wiring section (Standard Bus EIA-485 Communications).

2.2.2 Give the PMI a valid network address (IP and Subnet) using EZ-ZONE Configurator software or the PMI front panel. Configurator software is free of charge, if not already acquired, it can be downloaded from the Watlow website. (http://www.watlow.com/products/controllers/software.cfm)

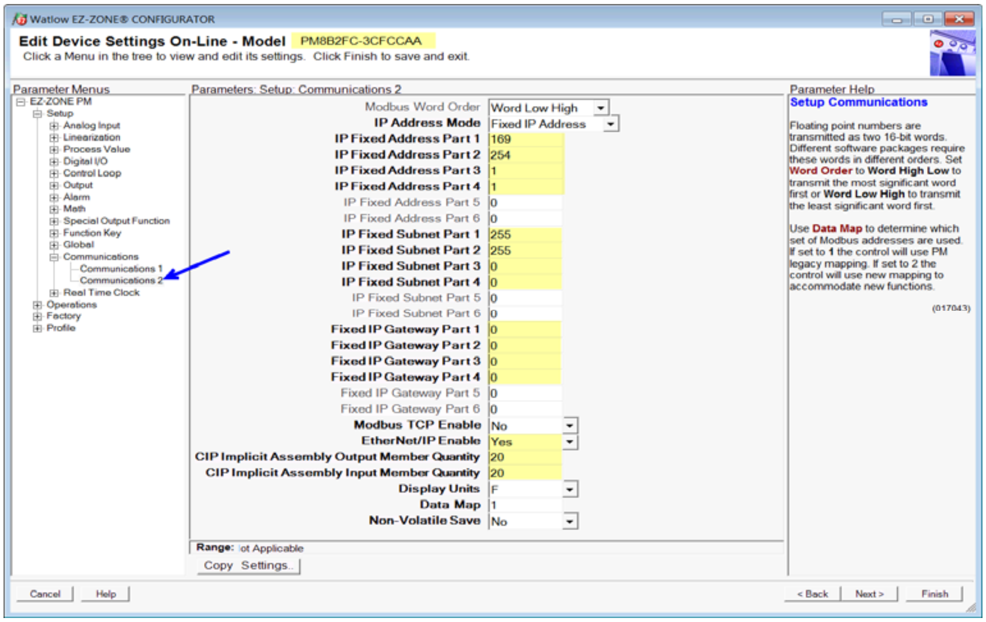
2.2.3 Double-click on the PMI or click on it once and the click the Next button to connect to the RUI.



2.2.4 Once connected, navigate to the Communications Menu and then to the Communications 2 parameter. Once there, enter the required network IP and Subnet address while also enabling EtherNet/IP (yellow highlight for emphasis).

Note:

After changing any part of the IP address power must be cycled for the change to take affect.



Note:

The input assembly within the PLC will always be set to n+1 where n = the size of the input assembly.

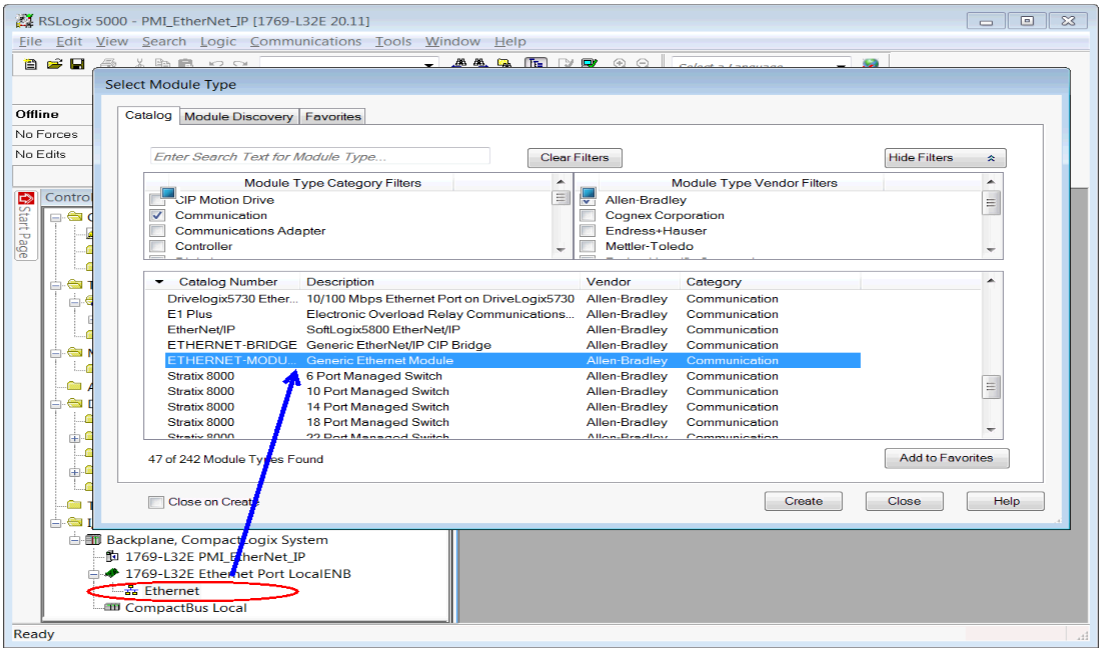
2.2.5 If explicit messaging will be used alone, the minimum assembly size requirement (from the PLC perspective) is 1 input and 1 output. Again, the assembly size must be the same in the PLC and the PMI. In the previous step, the graphic shows the assemblies set to twenty. *This was done now for implicit communication examples that will follow later in this document*.

2.3 PLC Configuration, Step-by-Step

2.3.1 Open RSLogix5000 software and add an additional I/O module. Follow the steps below to accomplish this task.

2.3.2 Navigate to the I/O Configuration folder structure. If not already expanded, do so now by clicking the plus sign next to it.

2.3.3 Right click the Ethernet port to add a new module. To narrow the search, select the "Communications" category under Module Type and Allen-Bradley under Module Vendor. Double-click on "Generic Ethernet Module".



2.3.4 Within the Generic Ethernet dialog, define the PMI properties. Fields that must be completed include (highlighted yellow in the graphic below):

Name: Given name becomes controller tags to be used in program.

Comm Format: Defines how data is to be treated within PLC.

Note:

All EZ-ZONE assembly members are 32-bits in length. If the Comm Format is set to something other than DINT, ensure the size changes in a corresponding fashion. As an example, if 20 (32-bit) members are in use, the appropriate Comm Formats would be:

DINT (32-bit): *Inputs = 21, Outputs = 20

INT (16-bit): *Inputs = 42, Outputs = 40

SINT (8-bit): *Inputs = 84, Outputs = 80

* The input assemblies within EZ-ZONE controllers have a dedicated Status member that is always present. The input assembly size will always be n+1 where n = the size of the input assembly.

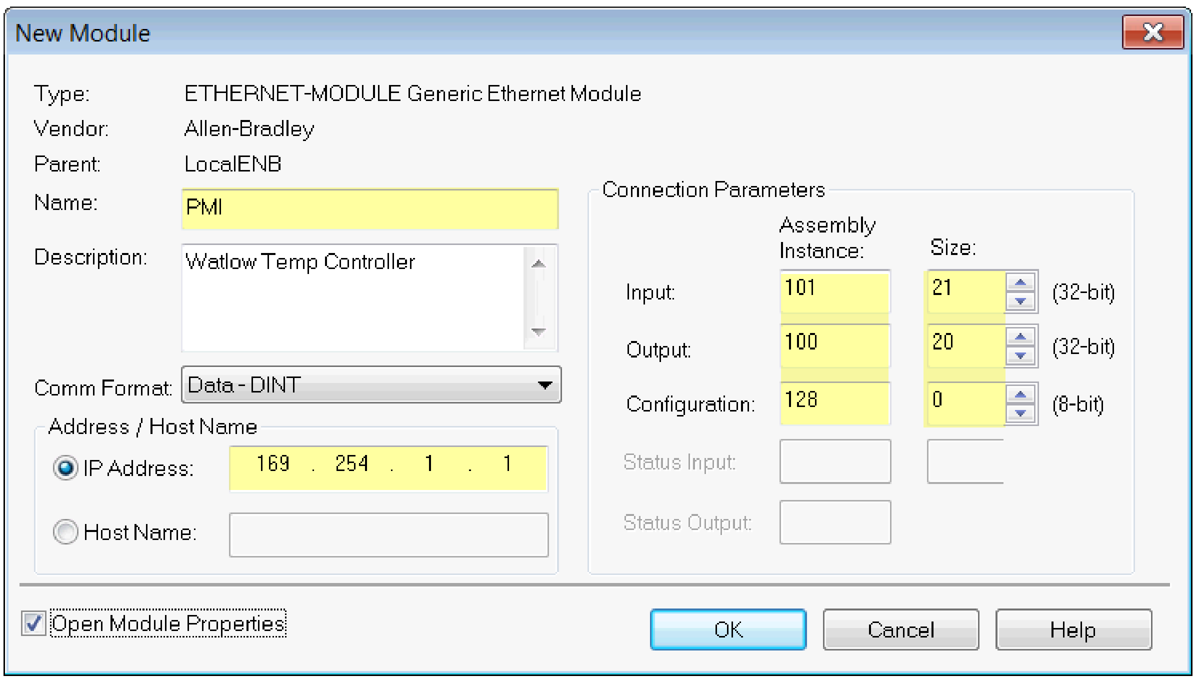
IP Address: Network PMI Ethernet address.

Assembly Instance

Input (101): Defines number of members to be included in the Input Implicit Assembly (as seen in EZ-ZONE Configurator software "*CIP Implicit Assembly* ***Output*** *Member Quantity*" from EZ-ZONE PMI).

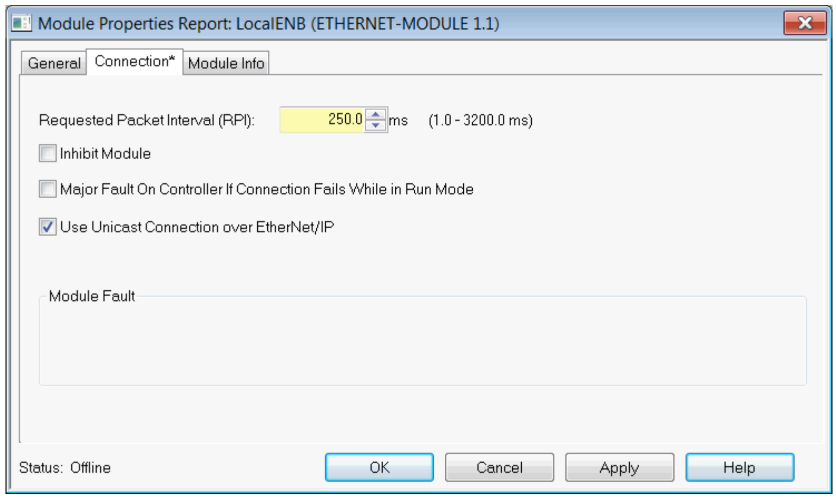
Output (100): Defines number of members to be included in the Output Implicit Assembly (as seen in EZ-ZONE Configurator software "*CIP Implicit Assembly* ***Input*** *Member Quantity"* from master devices).

Configuration (128): Always enter zero.



2.3.5 Click OK when all fields have been entered.

2.3.6 Enter the Requested packet interval, used with implicit messaging (250 to 500ms).

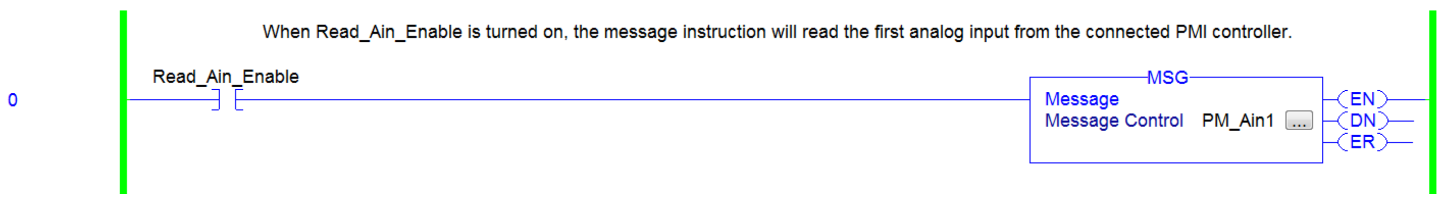


2.3.7 Click OK, the PMI configuration is now complete.

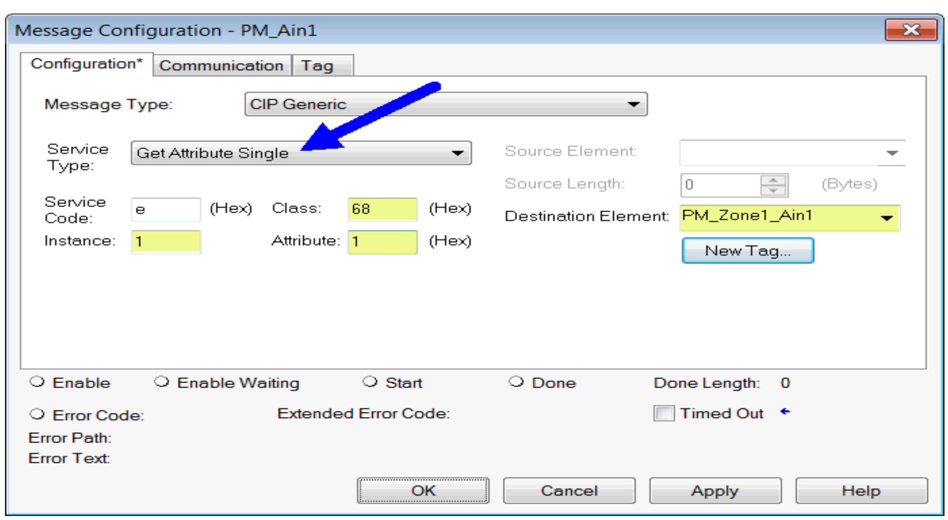
2.4 Explicit Programming Examples, Step-by-Step

The examples below will use a very simple and straight forward way to execute an explicit message. There are other ways to do this within the PLC.

2.4.1 The first example will read the first Analog input from a PM Integrated controller. To do this, create a rung of logic similar to that shown below.



2.4.2 Message instruction configuration with explanations.



Explanation of fields in graphic above follows.

- *Service Type*

This particular example will read a single parameter (attribute) from a PM controller, therefore the Service Type is "Get Attribute Single".

- *Class, Instance and Attribute*

Note:

The Class and Attribute are always entered in hexadecimal where the instance is entered in decimal.

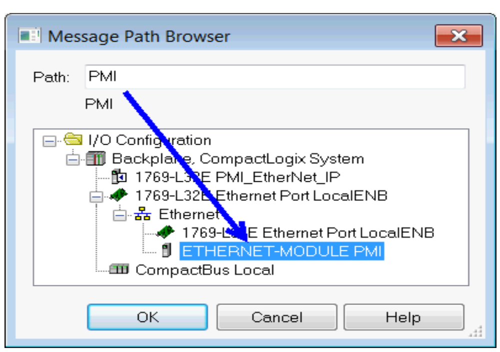
- *Destination Element*

The tag shown above (PM_Zone1_Ain1) must be created by the user and represents the tag within the PLC in which the Analog Input value will be found when the message instruction is executed.

The Class, Instance and Attribute as entered above, represents the actual address within the PM controller for the Analog Input. This address can be found in the PM Integrated (PMI) User's Guide within the Operations Page.

Operations Page								
Display	Parameter Name Description	Range	Default	Modbus Relative Address	CIP Class Instance Attribute hex (dec)	Profibus Index	Parameter ID	Data Type and Access **
Ai oPEr Analog Input Menu								
Ain	*Analog Input (1 to 2)* **Value** View the process value. **Note:** Ensure that the Error Status (below) indicates no error (61) when reading this value using a field bus protocol. If an error exists, the last known value prior to the error occurring will be returned.	-1,999.000 to 9,999.000°F or units -1,128.000 to 5,537.000°C	- - - -	***Instance 1*** Map 1 360, Map 2 360 ***Instance 2*** Map 1 440, Map 2 450	0x68 (104) 1 to 2 1	0	4001	float R

2.4.3 After completing the configuration tab of the message instruction, click the Communications tab to identify the path to the PM controller. Click the Browse button and select the PMI gateway as shown in the graphic below.



Lastly, click the OK button to finish the message instruction configuration.

2.4.4 Looking at the graphic in step 2.4.1 when the contact identified as "Read_Ain_Enable" comes on, the message instruction will be executed and the Analog Input will be read from the PM controller and then stored in the PLC tag called "PM_Zone1_Ain1".

2.5 Modifying Implicit Assemblies Using Explicit Messages, Step-by-Step

2.5.1 The first four default members of the PM Originator (PLC) to Target (EZ-ZONE controller) assembly is shown below.

PM - Originator (PLC Output Assembly Instance = 100) to Target (EZ-ZONE)					
Assembly Member	CIP - Assembly Member Address **Class, Instance, Attribute**	CIP - EZ-ZONE Parameter Adress **Class, Instance, Attritbute (Pointer)**	EZ-ZONE Parameter Name and Function (description)	Assembly Data Type	PLC Data Type
1	0x77, 0x01, 0x01	0x97, 0x01, 0x01	Control Loop 1, User Control Mode	DINT	DINT
2	0x77, 0x01, 0x02	0x6B, 0x01, 0x01	Control Loop 1, Closed Loop Set Point	DINT	REAL
3	0x77, 0x01, 0x03	0x6B, 0x01, 0x02	Control Loop 1, Open Loop Set Point	DINT	REAL
4	0x77, 0x01, 0x04	0x6D, 0x01, 0x01	Alarm 1, Alarm High Set Point	DINT	REAL

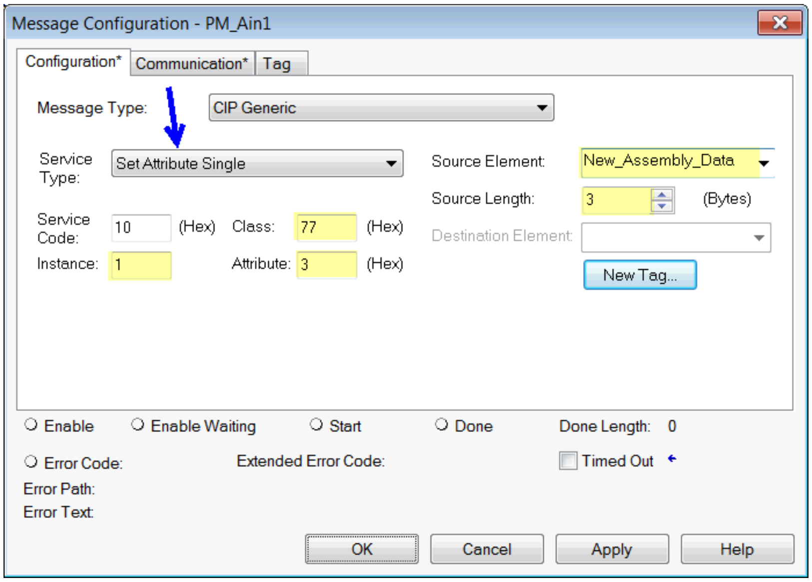
Note:

All numbers in the graphic above and in examples that follow that are preceded by 0x are in hexadecimal format; numbers without the prefix of 0x are in decimal format. The green highlight above indicates that this particular assembly (output) is instance one.

To change the 3rd Assembly Member from what is shown above to Control Mode loop 2, first find the appropriate CIP address in the PMI User's Guide (shown below).

LooP oPEr Control Loop Menu								
r.En r.En	*Control Loop (1 to 2)* **Remote Set Point** Enable this loop to switch control to the remote set point.	no No (59) YES Yes (106)	No	***Instance 1*** Map 1 Map 2 2200 2680 ***Instance 2*** Map 1 Map 2 2280 2760	0x6B (107) 1 to 2 0x15 (21)	48	7021	uint RWES
C.M C.M	*Control Loop (1 to 2)* **Control Mode** Select the method that this loop will use to control.	oFF Off (62) AUto Auto (10) MAn Manual (54)	Auto	***Instance 1*** Map 1 Map 2 1880 2360 ***Instance 2*** Map 1 Map 2 1950 2430	0x97 (151) 1 to 2 1	63	8001	uint RWES

The explicit message instruction configuration (previously discussed in step 2.4.2) now becomes a set (write) operation while a specific tag must be created which contains the new parameter address pointer (New_Assembly_Data) to be written to the designated assembly member. The message configuration would change as shown below.



Notice that the Source Element was created as a 3 dimensional array using the SINT data type because the Class, Instance and Attribute objects are 8-bits in length.

⊟ New_Assembly_Data	{...}	{...}	Hex	SINT[3]
⊞ New_Assembly_Data[0]	16#97		Hex	SINT
⊞ New_Assembly_Data[1]	2		Decimal	SINT
⊞ New_Assembly_Data[2]	16#01		Hex	SINT

3.0 Implicit Communications

3.1 PLC Configuration

Each PMI controller has a built-in implicit assembly. The I/O assemblies (shown below) reflect the factory defaults.

Note:

If the assemblies have been changed from the factory defaults it is important to know that if the controller is at any point thereafter brought back to factory default settings, the assemblies will be overwritten.

PM - Originator (PLC Output Assembly Instance = 100) to Target (EZ-ZONE)					
Assembly Member	CIP - Assembly Member Address **Class, Instance, Attribute**	CIP - EZ-ZONE Parameter Adress **Class, Instance, Attritbute (Pointer)**	EZ-ZONE Parameter Name and Function (description)	Assembly Data Type	PLC Data Type
1	0x77, 0x01, 0x01	0x97, 0x01, 0x01	Control Loop 1, User Control Mode	DINT	DINT
2	0x77, 0x01, 0x02	0x6B, 0x01, 0x01	Control Loop 1, Closed Loop Set Point	DINT	REAL
3	0x77, 0x01, 0x03	0x6B, 0x01, 0x02	Control Loop 1, Open Loop Set Point	DINT	REAL
4	0x77, 0x01, 0x04	0x6D, 0x01, 0x01	Alarm 1, Alarm High Set Point	DINT	REAL
5	0x77, 0x01, 0x05	0x6D, 0x01, 0x02	Alarm 1, Alarm Low Set Point	DINT	REAL
6	0x77, 0x01, 0x06	0x6D, 0x02, 0x01	Alarm 2, Alarm High Set Point	DINT	REAL
7	0x77, 0x01, 0x07	0x6D, 0x02, 0x02	Alarm 2, Alarm Low Set Point	DINT	REAL
8	0x77, 0x01, 0x08	0x6D, 0x03, 0x01	Alarm 3, Alarm High Set Point	DINT	REAL
9	0x77, 0x01, 0x09	0x6D, 0x03, 0x02	Alarm 3, Alarm Low Set Point	DINT	REAL
10	0x77, 0x01, 0x0A	0x6D, 0x04, 0x01	Alarm 4, Alarm High Set Point	DINT	REAL
11	0x77, 0x01, 0x0B	0x6D, 0x04, 0x02	Alarm 4 - Alarm Low Set Point	DINT	REAL
12	0x77, 0x01, 0x0C	0x7A, 0x01, 0x0B	Profile Action Request	DINT	DINT
13	0x77, 0x01, 0x0D	0x7A, 0x01, 0x01	Profile Start	DINT	DINT
14	0x77, 0x01, 0x0E	0x97, 0x01, 0x06	Control Loop 1, Heat Proportional Band	DINT	REAL
15	0x77, 0x01, 0x0F	0x97, 0x01, 0x07	Control Loop 1, Cool Proportional Band	DINT	REAL
16	0x77, 0x01, 0x10	0x97, 0x01, 0x08	Control Loop 1, Time Integral	DINT	REAL
17	0x77, 0x01, 0x11	0x97, 0x01, 0x09	Control Loop 1, Time Derivative	DINT	REAL
18	0x77, 0x01, 0x12	0x97, 0x01, 0x0B	Control Loop 1, Heat Hysteresis	DINT	REAL
19	0x77, 0x01, 0x13	0x97, 0x01, 0x0C	Control Loop 1, Cool Hysteresis	DINT	REAL
20	0x77, 0x01, 0x14	0x97, 0x01, 0x0A	Control Loop 1, Dead Band	DINT	REAL

PM - Target (EZ-ZONE) to Originator (PLC Input Assembly Instance = 101)					
Assembly Member	CIP - Assembly Member Address **Class, Instance, Attribute**	CIP - EZ-ZONE Parameter Adress **Class, Instance, Attritbute (Pointer)**	EZ-ZONE Parameter Name and Function (description)	Assembly Data Type	PLC Data Type
0	**none**	**none**	**Device Status**	**DINT**	**BIN**
1	0x77, 0x02, 0x01	0x68, 0x01, 0x01	Analog Input 1, Analog Input Value	DINT	REAL
2	0x77, 0x02, 0x02	0x68, 0x01. 0x02	Analog Input 1, Input Error	DINT	REAL
3	0x77, 0x02, 0x03	0x68, 0x02, 0x01	Analog Input 2, Analog Input Value	DINT	REAL
4	0x77, 0x02, 0x04	0x68, 0x02, 0x02	Analog Input 2, Input Error	DINT	REAL
5	0x77, 0x02, 0x05	0x6D, 0x01, 0x09	Alarm 1, Alarm State	DINT	DINT
6	0x77, 0x02, 0x06	0x6D, 0x02, 0x09	Alarm 2, Alarm State	DINT	DINT
7	0x77, 0x02, 0x07	0x6D, 0x03, 0x09	Alarm 3, Alarm State	DINT	DINT
8	0x77, 0x02, 0x08	0x6D, 0x04, 0x09	Alarm 4, Alarm State	DINT	DINT
9	0x77, 0x02, 0x09	0x6E, 0x01, 0x05	Digital Input 1, Event Status	DINT	DINT
10	0x77, 0x02, 0x0A	0x6E, 0x02, 0x05	Digital Input 2, Event Status	DINT	DINT
11	0x77, 0x02, 0x0B	0x97, 0x01, 0x02	Control Loop 1, Control Mode Active	DINT	DINT
12	0x77, 0x02, 0x0C	0x97, 0x01, 0x0D	Control Loop 1, Heat Power	DINT	REAL
13	0x77, 0x02, 0x0D	0x97, 0x01, 0x0E	Control Loop 1, Cool Power	DINT	REAL
14	0x77, 0x02, 0x0E	0x70, 0x01, 0x06	Limit State	DINT	DINT
15	0x77, 0x02, 0x0F	0x7A, 0x01, 0x01	Profile Start	DINT	DINT
16	0x77, 0x02, 0x10	0x7A, 0x01, 0x0B	Profile Action Request	DINT	DINT
17	0x77, 0x02, 0x11	0x7A, 0x01, 0x03	Current Profile	DINT	DINT
18	0x77, 0x02, 0x12	0x7A, 0x01, 0x04	Current Step	DINT	DINT
19	0x77, 0x02, 0x13	0x7A, 0x01, 0x05	Profile Active Set Point	DINT	REAL
20	0x77, 0x02, 0x14	0x7A, 0x01, 0x09	Step Time Remaining	DINT	DINT

3.1.1 In step 2.2.5 above, the Implicit Assemblies (input and output) were configured for 20 members. With the PMI configuration complete, it is time to look closer at the PLC side.

3.1.2 In step 2.3.4 above, the PLC was configured to include a Generic Ethernet module. The module parameters, specifically, the assembly sizes were configured at this time to be the same as the PMI (20 in and 20 out).

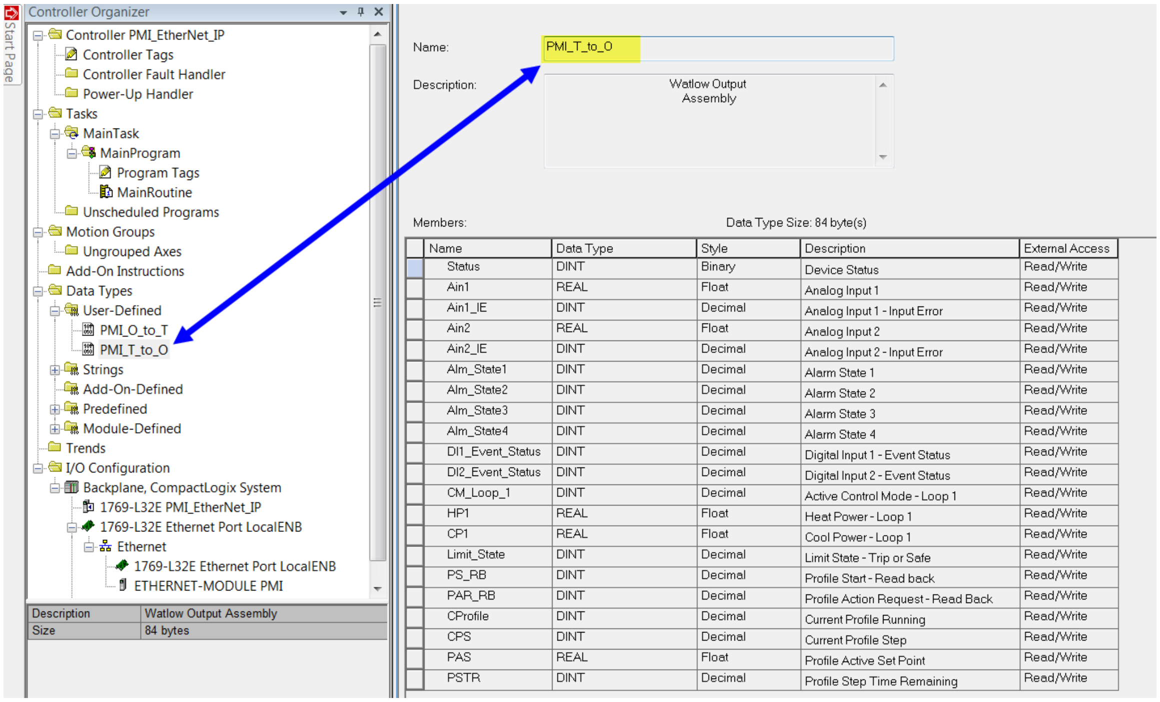
3.1.3 Prior to making any changes to the current PLC configuration let's look closer at what is currently there. Recall that when a Generic Ethernet module was added to the PLC I/O structure (step 2.3.4) it was also given a name (PMI) and that name became a controller tag. While being connected to the PLC on-line the input tag (PMI:I) clearly shows that it is dynamically receiving raw data from the Slave (see the screenshot below).

Name	Value	Style	Data Type
PMI:I	{...}		AB:ETHERNE...
PMI:I.Data	{...}	Decimal	DINT[21]
PMI:I.Data[0]	69632	Decimal	DINT
PMI:I.Data[1]	1117721581	Decimal	DINT
PMI:I.Data[2]	61	Decimal	DINT
PMI:I.Data[3]	1117371871	Decimal	DINT
PMI:I.Data[4]	61	Decimal	DINT
PMI:I.Data[5]	88	Decimal	DINT
PMI:I.Data[6]	88	Decimal	DINT
PMI:I.Data[7]	88	Decimal	DINT
PMI:I.Data[8]	88	Decimal	DINT
PMI:I.Data[9]	41	Decimal	DINT
PMI:I.Data[10]	41	Decimal	DINT
PMI:I.Data[11]	10	Decimal	DINT
PMI:I.Data[12]	0	Decimal	DINT
PMI:I.Data[13]	0	Decimal	DINT
PMI:I.Data[14]	0	Decimal	DINT
PMI:I.Data[15]	1	Decimal	DINT
PMI:I.Data[16]	61	Decimal	DINT
PMI:I.Data[17]	0	Decimal	DINT
PMI:I.Data[18]	0	Decimal	DINT
PMI:I.Data[19]	0	Decimal	DINT
PMI:I.Data[20]	0	Decimal	DINT

3.2 PLC Programming

Being that the graphic above shows the T to O assembly, 21 members are present. Some of the values coming in reflect numbers that are expected (members [0], [2] and [4]) but some do not (members [1] and [3] due to the data format shown and the data format of the given parameter.

3.2.1 To start the programming process, it is suggested that a User Defined Data Type be created for both implicit assemblies which will reflect the appropriate data format and will also simplify the programming when transferring I/O data between Master (PLC) and Slave (PMI module). As can be seen below, a user defined data type was created (PMI_T_to_O) using the PMI default assembly. Also notice that the first member is defined as the Device Status.

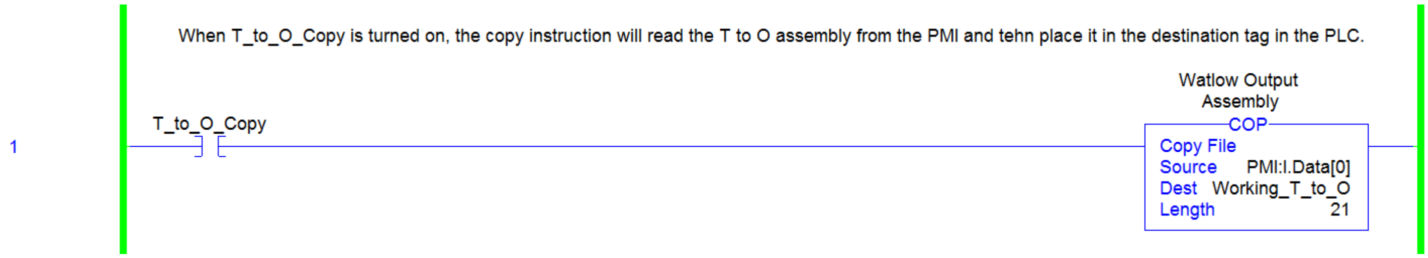


3.2.2 Once created, a controller tag should be created using this User Defined Data Type as its data type. Below, a controller tag was created (Working_T_to_O) where the data type (yellow highlight) is the User Defined Data Type created in the previous step.

Name	Data Type	Description
Working_T_to_O	PMI_T_to_O	Watlow Output Assembly
Working_T_to_O.Status	DINT	Watlow Output Assembly Device Status
Working_T_to_O.Ain1	REAL	Watlow Output Assembly Analog Input 1
Working_T_to_O.Ain1_IE	DINT	Watlow Output Assembly Analog Input 1 - Input Error
Working_T_to_O.Ain2	REAL	Watlow Output Assembly Analog Input 2
Working_T_to_O.Ain2_IE	DINT	Watlow Output Assembly Analog Input 2 - Input Error
Working_T_to_O.Alm_State1	DINT	Watlow Output Assembly Alarm State 1
Working_T_to_O.Alm_State2	DINT	Watlow Output Assembly Alarm State 2
Working_T_to_O.Alm_State3	DINT	Watlow Output Assembly Alarm State 3
Working_T_to_O.Alm_State4	DINT	Watlow Output Assembly Alarm State 4
Working_T_to_O.DI1_Event_Status	DINT	Watlow Output Assembly Digital Input 1 - Event Status
Working_T_to_O.DI2_Event_Status	DINT	Watlow Output Assembly Digital Input 2 - Event Status
Working_T_to_O.CM_Loop_1	DINT	Watlow Output Assembly Active Control Mode - Loop 1
Working_T_to_O.HP1	REAL	Watlow Output Assembly Heat Power - Loop 1
Working_T_to_O.CP1	REAL	Watlow Output Assembly Cool Power - Loop 1
Working_T_to_O.Limit_State	DINT	Watlow Output Assembly Limit State - Trip or Safe
Working_T_to_O.PS_RB	DINT	Watlow Output Assembly Profile Start - Read back
Working_T_to_O.PAR_RB	DINT	Watlow Output Assembly Profile Action Request - Read Back
Working_T_to_O.CProfile	DINT	Watlow Output Assembly Current Profile Running
Working_T_to_O.CPS	DINT	Watlow Output Assembly Current Profile Step
Working_T_to_O.PAS	REAL	Watlow Output Assembly Profile Active Set Point
Working_T_to_O.PSTR	DINT	Watlow Output Assembly Profile Step Time Remaining

The example above was based on the default PMI T to O assembly. This assembly will typically be modified by each user and would look different based on each assembly configuration and the parameters in use.

3.2.3 Enter the rung of logic shown below to write the Slave (PM controller) data into the Master (PLC) tag created above.



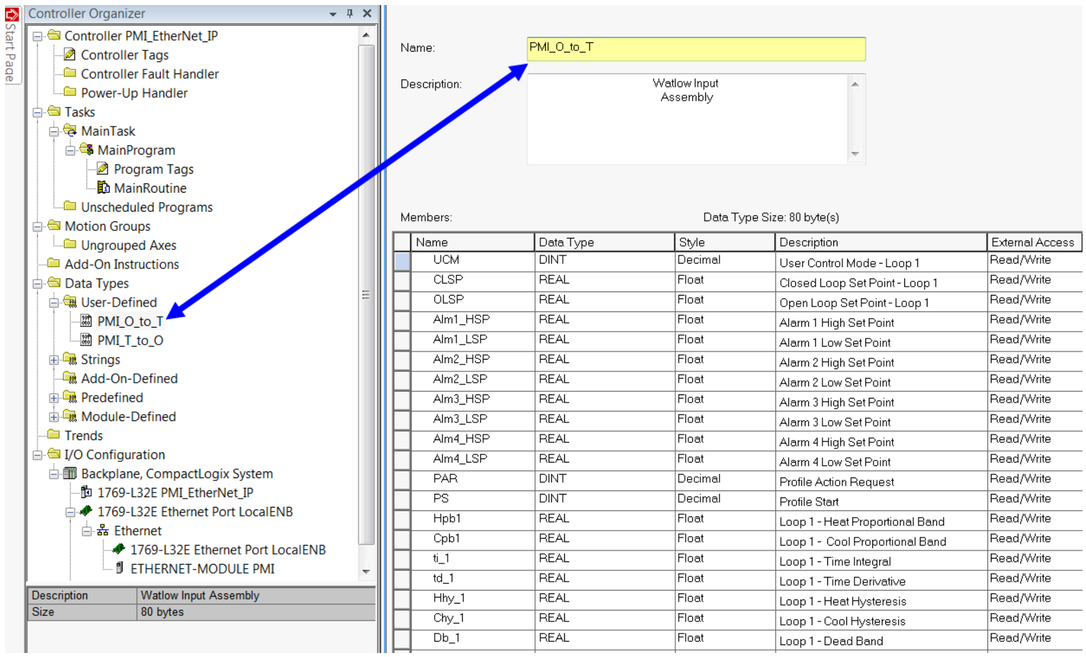
Notice that the source of the copy instruction has the same name as the name given to the module back in step 2.3.4. Recall that when the module was added that there were entries for the input, output and configuration assemblies. The input assembly was defined as having 21 members as was the destination tag. Therefore, the length is defined as 21. Once the contact (T_to_O_Copy) is enabled, the source data will be copied to the destination PLC tag as can be seen below.

Name	Value	Style	Data Type
Working_T_to_O	{...}		PMI_T_to_O
Working_T_to_O.Status	2#0000_0000_0000_0001_0001_0000_0000_0000	Binary	DINT
Working_T_to_O.Ain1	83.069855	Float	REAL
Working_T_to_O.Ain1_IE	61	Decimal	DINT
Working_T_to_O.Ain2	79.012856	Float	REAL
Working_T_to_O.Ain2_IE	61	Decimal	DINT
Working_T_to_O.Alm_St...	88	Decimal	DINT
Working_T_to_O.Alm_St...	88	Decimal	DINT
Working_T_to_O.Alm_St...	88	Decimal	DINT
Working_T_to_O.Alm_St...	88	Decimal	DINT
Working_T_to_O.DI1_E...	41	Decimal	DINT
Working_T_to_O.DI2_E...	41	Decimal	DINT
Working_T_to_O.CM_Lo...	10	Decimal	DINT
Working_T_to_O.HP1	0.0	Float	REAL
Working_T_to_O.CP1	0.0	Float	REAL
Working_T_to_O.Limit_...	0	Decimal	DINT
Working_T_to_O.PS_RB	1	Decimal	DINT
Working_T_to_O.PAR_...	61	Decimal	DINT
Working_T_to_O.CProfile	0	Decimal	DINT
Working_T_to_O.CPS	0	Decimal	DINT
Working_T_to_O.PAS	0.0	Float	REAL
Working_T_to_O.PSTR	0	Decimal	DINT

Now that the data formats correspond to each parameter (member) data type, we see values that are more in alignment with expectations.

Again, looking at the graphic above, notice the first member referred to as "Device Status". For a PMI controller the bits shown as set to a "1" will always be set indicating valid communications with the Master.

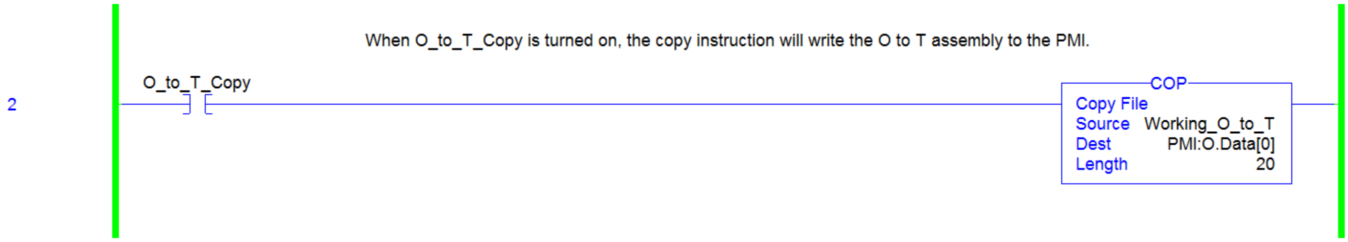
3.2.4 To write data out to the Target from the Master, perform the same steps in the creation of the O to T assembly; as can be seen below, the User Defined Data Type is created based on the PMI defaults. Keep in mind that this assembly represents the output from the PLC (Originator) which will be sent to the Target (PMI). The program within the PLC would write values to these tags and they would then be sent out to the Target at the rate of the setting for the RPI (see step 2.3.6).



3.2.5 The corresponding O to T controller tag is created as shown below.

Name	Data Type	Description
Working_O_to_T	PMI_O_to_T	Watlow Input Assembly
Working_O_to_T.UCM	DINT	Watlow Input Assembly User Control Mode - Loop 1
Working_O_to_T.CLSP	REAL	Watlow Input Assembly Closed Loop Set Point - Loop 1
Working_O_to_T.OLSP	REAL	Watlow Input Assembly Open Loop Set Point - Loop 1
Working_O_to_T.Alm1_HSP	REAL	Watlow Input Assembly Alarm 1 High Set Point
Working_O_to_T.Alm1_LSP	REAL	Watlow Input Assembly Alarm 1 Low Set Point
Working_O_to_T.Alm2_HSP	REAL	Watlow Input Assembly Alarm 2 High Set Point
Working_O_to_T.Alm2_LSP	REAL	Watlow Input Assembly Alarm 2 Low Set Point
Working_O_to_T.Alm3_HSP	REAL	Watlow Input Assembly Alarm 3 High Set Point
Working_O_to_T.Alm3_LSP	REAL	Watlow Input Assembly Alarm 3 Low Set Point
Working_O_to_T.Alm4_HSP	REAL	Watlow Input Assembly Alarm 4 High Set Point
Working_O_to_T.Alm4_LSP	REAL	Watlow Input Assembly Alarm 4 Low Set Point
Working_O_to_T.PAR	DINT	Watlow Input Assembly Profile Action Request
Working_O_to_T.PS	DINT	Watlow Input Assembly Profile Start
Working_O_to_T.Hpb1	REAL	Watlow Input Assembly Loop 1 - Heat Proportional Band
Working_O_to_T.Cpb1	REAL	Watlow Input Assembly Loop 1 - Cool Proportional Band
Working_O_to_T.ti_1	REAL	Watlow Input Assembly Loop 1 - Time Integral
Working_O_to_T.td_1	REAL	Watlow Input Assembly Loop 1 - Time Derivative
Working_O_to_T.Hhy_1	REAL	Watlow Input Assembly Loop 1 - Heat Hysteresis
Working_O_to_T.Chy_1	REAL	Watlow Input Assembly Loop 1 - Cool Hysteresis
Working_O_to_T.Db_1	REAL	Watlow Input Assembly Loop 1 - Dead Band

3.2.6 Enter the rung of logic shown below to write data from the Originator to the Target.



Notice that the source of the copy instruction is now the controller tag created above where the destination is the same name given to the module back in step 2.3.4. In this case, the output assembly was defined as having 20 members as was the destination tag. Therefore, the length is defined as 20. Once the contact (O_to_T_Copy) is enable the source data will be sent to the destination as can be seen below.

⊟ Working_O_to_T	{...}		PMI_O_to_T	Watlow Input Assembly
⊞ Working_O_to_T.UCM	10	Decimal	DINT	Watlow Input Assembly User Control Mode - Loop 1
Working_O_to_T.CLSP	250.0	Float	REAL	Watlow Input Assembly Closed Loop Set Point - Loop 1
Working_O_to_T.OLSP	45.0	Float	REAL	Watlow Input Assembly Open Loop Set Point - Loop 1
Working_O_to_T.Alm1_HSP	300.0	Float	REAL	Watlow Input Assembly Alarm 1 High Set Point
Working_O_to_T.Alm1_LSP	175.0	Float	REAL	Watlow Input Assembly Alarm 1 Low Set Point
Working_O_to_T.Alm2_HSP	250.0	Float	REAL	Watlow Input Assembly Alarm 2 High Set Point
Working_O_to_T.Alm2_LSP	150.0	Float	REAL	Watlow Input Assembly Alarm 2 Low Set Point
Working_O_to_T.Alm3_HSP	200.0	Float	REAL	Watlow Input Assembly Alarm 3 High Set Point
Working_O_to_T.Alm3_LSP	125.0	Float	REAL	Watlow Input Assembly Alarm 3 Low Set Point
Working_O_to_T.Alm4_HSP	175.0	Float	REAL	Watlow Input Assembly Alarm 4 High Set Point
Working_O_to_T.Alm4_LSP	100.0	Float	REAL	Watlow Input Assembly Alarm 4 Low Set Point
⊞ Working_O_to_T.PAR	0	Decimal	DINT	Watlow Input Assembly Profile Action Request
⊞ Working_O_to_T.PS	0	Decimal	DINT	Watlow Input Assembly Profile Start
Working_O_to_T.Hpb1	8.0	Float	REAL	Watlow Input Assembly Loop 1 - Heat Proportional Band
Working_O_to_T.Cpb1	6.0	Float	REAL	Watlow Input Assembly Loop 1 - Cool Proportional Band
Working_O_to_T.ti_1	2.0	Float	REAL	Watlow Input Assembly Loop 1 - Time Integral
Working_O_to_T.td_1	3.0	Float	REAL	Watlow Input Assembly Loop 1 - Time Derivative
Working_O_to_T.Hhy_1	3.0	Float	REAL	Watlow Input Assembly Loop 1 - Heat Hysteresis
Working_O_to_T.Chy_1	3.0	Float	REAL	Watlow Input Assembly Loop 1 - Cool Hysteresis
Working_O_to_T.Db_1	5.0	Float	REAL	Watlow Input Assembly Loop 1 - Dead Band

Note:

Due to the fact that the default O to T assembly has the PID parameters in member locations 14 - 17 (yellow highlight above) the controller autotune feature will be immediately overwritten once complete. Therefore, it is suggested that these member locations be changed prior to performing an autotune as described in section 2.5.